

# Examen MercadoLibre

# Tecnologías utilizadas

## ¿Por qué NodeJS?

Se eligió JavaScript (NodeJS) para el desarrollo de esta aplicación, ya que no se poseen objetivos: de seguridad, performance o stress. Opté por esta tecnología que permite realizar una API en poco tiempo, gracias a las librerías (siempre intentando minimizar su presencia para que el código sea lo más limpio posible) y la facilidad de desarrollar el código.

## ¿Que base de datos utilicé?

Opté por utilizar MySQL, por la facilidad de implementación. También realice un apoyo en Redis para almacenar cache, ya que al entrar en el contexto de ser una API consumida por una o más civilizaciones tal vez Deba realizar en días particulares el mismo request repetido y el cache en ese caso daría un alivio a las llamadas a la base de datos.

Se debe tener en cuenta que en el servidor AWS el redis se encuentra consumido por dockers, uno actuando como slave(read-only) y otro como master(todas las operaciones).

## Scripts

A continuación se copian los script de inicialización de la base de datos:

```
1
2 CREATE TABLE `calendar` (
3   `id` int(11) NOT NULL,
4   `p1x` int(11) NOT NULL,
5   `p1y` int(11) NOT NULL,
6   `p2x` int(11) NOT NULL,
7   `p2y` int(11) NOT NULL,
8   `p3x` int(11) NOT NULL,
9   `p3y` int(11) NOT NULL
10  ) ENGINE=InnoDB DEFAULT CHARSET=utf8 COLLATE=utf8_unicode_ci;
11
12 ALTER TABLE `calendar`
13   ADD PRIMARY KEY (`id`);
14
15 ALTER TABLE `calendar`
16   MODIFY `id` int(11) NOT NULL AUTO_INCREMENT;
17
18
```

Primeros 50 elementos generados:

```
mysql> select * from calendar;
```

id	p1x	p1y	p2x	p2y	p3x	p3y
1	500	-9	1997	-105	996	87
2	500	-17	1989	-209	985	174
3	499	-26	1975	-313	966	259
4	499	-35	1956	-416	940	342
5	498	-44	1932	-518	906	423
6	497	-52	1902	-618	866	500
7	496	-61	1867	-717	819	574
8	495	-70	1827	-813	766	643
9	494	-78	1782	-908	707	707
10	492	-87	1732	-1000	643	766
11	491	-95	1677	-1089	574	819
12	489	-104	1618	-1176	500	866
13	487	-112	1554	-1259	423	906
14	485	-121	1486	-1338	342	940
15	483	-129	1414	-1414	259	966
16	481	-138	1338	-1486	174	985
17	478	-146	1259	-1554	87	996
18	476	-155	1176	-1618	0	1000
19	473	-163	1089	-1677	-87	996
20	470	-171	1000	-1732	-174	985
21	467	-179	908	-1782	-259	966
22	464	-187	813	-1827	-342	940
23	460	-195	717	-1867	-423	906
24	457	-203	618	-1902	-500	866
25	453	-211	518	-1932	-574	819
26	449	-219	416	-1956	-643	766
27	446	-227	313	-1975	-707	707
28	441	-235	209	-1989	-766	643
29	437	-242	105	-1997	-819	574
30	433	-250	0	-2000	-866	500
31	429	-258	-105	-1997	-906	423
32	424	-265	-209	-1989	-940	342
33	419	-272	-313	-1975	-966	259
34	415	-280	-416	-1956	-985	174
35	410	-287	-518	-1932	-996	87
36	405	-294	-618	-1902	-1000	0
37	399	-301	-717	-1867	-996	-87
38	394	-308	-813	-1827	-985	-174
39	389	-315	-908	-1782	-966	-259
40	383	-321	-1000	-1732	-940	-342
41	377	-328	-1089	-1677	-906	-423
42	372	-335	-1176	-1618	-866	-500
43	366	-341	-1259	-1554	-819	-574
44	360	-347	-1338	-1486	-766	-643
45	354	-354	-1414	-1414	-707	-707
46	347	-360	-1486	-1338	-643	-766
47	341	-366	-1554	-1259	-574	-819
48	335	-372	-1618	-1176	-500	-866
49	328	-377	-1677	-1089	-423	-906
50	321	-383	-1732	-1000	-342	-940

## Librerías

Las librerías utilizadas para el desarrollo de la app son las siguientes:

"express": "\*", para levantar la API.

"body-parser": "\*", para usar el formato JSON en la aplicación.

"ioredis": "^3.2.2", para la conexión con Redis.

"mysql2": "^1.5.3", para la conexión con la base de datos.

"nodemon": "^1.18.9": para facilitar el desarrollo sin tener que reiniciar el servidor (solo dev).

## Estructura de clases

Para realizar las clases que compondrán la resolución del problema se genera una clase: connections que maneja las conexiones a MySQL y Redis según corresponda.

Además, se tiene una clase astronómico con los atributos de nombre y posición, para poder instanciar la referencia y también extender a una clase llamada planeta que además cuenta con los atributos: velocidad, sentido y radio. Se debe tener en cuenta que para las medidas de días y años se tomaron los días y años terrestres y no los días que cada planeta tarda en dar una vuelta al sol lo que sería otra forma de encarar el problema.

## Planteo del problema

Se decide dar al problema un enfoque simple para disminuir la cantidad de operaciones y dar una respuesta rápida ante peticiones.

Partiendo de la base que todo conjunto de 3 puntos con distinto radio forma un triángulo y si ese triángulo tiene área nula estamos en presencia de una recta. Ya podemos dividir en dos grupos el problema: lluvia ( $\text{area} > 0$ ) y sequía - óptima ( $\text{área} = 0$ ).

### Lluvia

Una vez que se verifica que la posición de los planetas forman un triángulo de área no nula, se realizan 3 productos vectoriales de cada una de las rectas, contra el punto de referencia (estrella): si ambos tienen el mismo signo, significa que el astro es contenido por el triángulo formado por los planetas.

### Sequía

Para que el día sea seco, ya sabiendo que estamos en presencia de una recta, debemos verificar si la recta formada por los planetas contiene al punto de referencia, por lo que si dos puntos tienen el mismo ángulo ya sabemos que estamos en presencia de un estado de sequía (tener en cuenta que uno de los 3 planetas puede tener un desfase de  $\pi$ , pero siempre al menos 2 ángulos deben ser iguales con respecto a la referencia).

## Óptimo

Si es una recta (área nula) y no contiene el punto de origen sin duda estamos en presencia de un estado óptimo de presión y temperatura.

## Endpoints

### start

Genera un registro de los siguientes 10 años en base comenzando por el día de la realización de este proyecto (02/01/2019) contando años bisiesto. Por motivos de alcance del proyecto, la respuesta es 'Success' o 'Error' según corresponda. La generación se produce en modo de transacción para evitar un grabado parcial en la base de datos.

### stats

Devuelve la respuesta esperada en el proyecto en formato JSON:

```
1 {  
2   "lluvia": 1224,  
3   "lluvia_max": 3501,  
4   "lluvia_max_dia": 23,  
5   "sequia": 40,  
6   "optimo": 0  
7 }  
8  
9
```

Como se puede ver de los 3652 días, 1224 son días lluviosos; 40 con sequías, en el cual hay un pico en el día 23, aunque no es el único ya que todo triángulo llega a su perímetro máximo cuando los ángulos sus lados son de 60 grados formando un triángulo equilátero. Días de óptima temperatura y presión no se dan en este caso ya que se tomó la decisión de guardar los datos como enteros ya que el ejercicio no pedía un formato y esto acota el universo de posibilidades donde se pueden dar los días de este tipo. Si se quieren realizar el proyecto con una mayor precisión se puede realizar un ALTER TABLE a las columnas de posición hacia un tipo float.

### clima

Parámetro: ?dias=:id

tipo de response del tipo success:

```
1 {  
2   "dia": 90,  
3   "clima": "sequia"  
4 }  
5  
6
```

En este caso se devuelve al usuario de la api, el día que solicitó junto con el tipo de día calculado por la app.

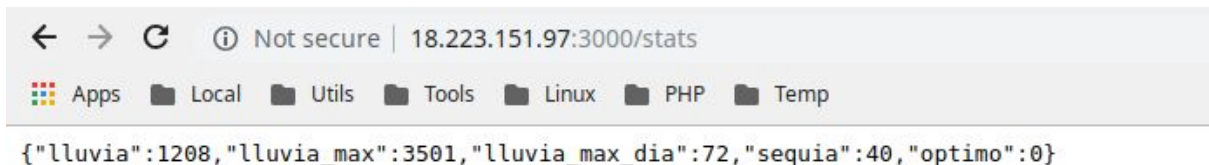
## Job

Se e parte del pedido de realizar realizar un Job como cliente para obtener todos los datos de 10 años adelante a la fecha, para eso se realiza un script en bash que consulta a la base de datos y guarda la información en un archivo '/Docs/data.txt' en este caso a modo de ejemplo.

```
1 #!/bin/bash  
2 set -B # enable brace expansion  
3 for i in {1..3652}; do  
4   curl -k 'GET' 'http://localhost:3000/clima?dias=$i' >> ./Docs/data.txt  
5   echo -e "\n" >> hola.txt  
6 done
```

## Deploy

Para finalizar se sube el servidor a un AWS de amazon para que pueda ser consultado:



The screenshot shows a web browser window with the address bar displaying "18.223.151.97:3000/stats". The browser's navigation bar includes back, forward, and refresh buttons, along with a "Not secure" warning. Below the address bar, there is a breadcrumb trail: "Apps > Local > Utils > Tools > Linux > PHP > Temp". The main content area of the browser displays a JSON object: {"lluvia":1208,"lluvia\_max":3501,"lluvia\_max\_dia":72,"sequia":40,"optimo":0}.

```
← → ↻ ⓘ Not secure | 18.223.151.97:3000/stats  
Apps Local Utils Tools Linux PHP Temp  
{"lluvia":1208,"lluvia_max":3501,"lluvia_max_dia":72,"sequia":40,"optimo":0}
```